A Decomposed Fourier-Motzkin Elimination Framework to Derive Capacity Models of Container Vessels

Mai L. Ajspur[1], Rune M. Jensen[1], and and Kent H. Andersen[2]

[1] The IT University of Copenhagen
[2] Aarhus University

Abstract. Accurate capacity models expressing the trade-off between different container types that can be stowed on container vessels are required in core liner shipping functions such as uptake, capacity, and network management. Today, simple models based on volume, weight, and refrigerated container capacity are used for these tasks, which causes overestimations that hamper decision making. Though previous work on stowage planning optimization in principle provide fine-grained linear Vessel Stowage Models (VSMs), they are too complex to be used in the mentioned functions. As an alternative, this paper contributes a novel framework based on Fourier-Motzkin elimination that automatically derives Vessel Capacity Models (VCMs) from VSMs by projecting unneeded variables. Our results show that the projected VCMs are reduced by an order of magnitude and can be solved 20-35 times faster than their corresponding VSMs with only a negligible loss in accuracy. Our framework is applicable to LP models in general, but are particularly effective on block-angular structured problems such as VSMs. We show similar results for a multi-commodity flow problem.

Keywords: Fourier-Motzkin Elimination · Capacity Model · Liner Shipping · Projection.

1 Introduction

Container shipping is a central element in the clockwork of global trade. A container liner shipping company operates a set of container vessels which sail on closed loop services with fixed schedules. These services connect major trade regions like Asia and Europe, and the liner shipping business is focused on utilizing the cargo capacity in their service network. Unused capacity constitute a loss that can be fatal in a market with a profit margin of just a few percent. To maximize utilization of capacity, it is central for a liner shipping company to be able to estimate the residual capacity of a container vessel. This is challenging in practice, since an empty slot may be impossible to utilize for a wide range of reasons including various stowage rules and seaworthiness requirements interact such that the free capacity of each container type and weight class is a complex function of the composition of cargo on board and the design of the vessel. Often

it is only the stowage planning team that can determine the residual capacity of a vessel accurately and usually they must spent hours planning the vessel manually to do it.

While stowage planners may be able to estimate free vessel capacity, the knowledge is primarily needed in higher functions such as: *uptake management* that control the sale of cargo bookings to fill the vessels with profitable cargo; *capacity management* that route cargo through the service network; and *network management* that makes changes to the service network. Decision makers in these functions seldom have stowage insight or time to consult the stowage team. They usually boil down the free capacity of a vessel to its nominal volume, weight, and reefer (refrigerated containers) capacity minus total volume, weight and reefer number of containers already on board. This simple three dimensional capacity model is inherently optimistic, since it ignores stowage complications. It has been shown that it can lead to revenue overestimates of more than 15% [5]. This can cause sub-optimal decisions that significantly harm business. Previous work has contributed frameworks for automated stowage planning (e.g.,[21, 12, 2, 16], and recently, linear stowage planning models were shown to scale to large container vessels (e.g., [5]). These latter Vessel Stowage Models (VSMs) embed an accurate capacity model, but since they include positioning information about the containers, they are too large for use as capacity models in higher functions, since these tasks often require several hundred capacity models to be solved simultaneously.

In this paper, we introduce a novel method to calculate a Vessel Capacity Model (VCM) automatically from a linear Vessel Stowage Model (VSM). Our basic idea is to derive the VCM by projecting out positioning variables from the VSM such that the VCM only expresses the relationship between variables that represent the total amount of each possible container type. Despite a theoretical double-exponential complexity, Fourier-Motzkin Elimination (FME) has been applied successfully for this purpose in previous work including constraint programming (e.g., [13]) and software verification (e.g., [4]). Our main contribution is to improve the state-of-the-art of these FME-based projection frameworks (e.g., [19, 14, 18]). In particular, we introduce a novel decomposition method that takes advantage of block-angular structured models such as VSMs to significantly speed-up the projection of variables in these models. Additionally, our removal of redundant constraints is parallellized, and the framework includes preprocessing of the constraint system including removal of less strict inequalities.

Our experimental evaluation of computing VCMs with this method shows that the number of constraints and non-zeros in the resulting VCMs typically are reduced by an order of magnitude compared to their corresponding VSMs. Moreover, the decomposition method reduces the size of the intermediary models produced by FME, causing less time to be needed for removing redundant constraints, which again speeds up the projection process significantly. In addition, for the models that include hydrostatic constraints, the resulting VCMs can be solved 20-35 times faster than their VSMs with only a negligible loss in accuracy. Although it can take several hours to derive a capacity model due to

the clean-up of redundant constraints, this only has to be done one time for a vessel class. In this way, the approach is suitable for computing capacity models. Moreover, since they are linear and much faster to solve than their corresponding stowage models, they also can be integrated in decision support systems used to optimise these higher functions in liner shipping. Multi-commodity flow problems also have a block-angular structure and we found a speed-up and a reduction in final size similar to the ones seen for VCMs.

This paper is organized as follows. Section 2 briefly presents the VSMs that are projected, and Section 3 introduces the required definitions and notation for our FME framework. Then Section 4 outlines the methods used in our FME framework including how block-angular problems are decomposed. Our experimental results then follows in Section 5. Section 6 present related work, before Section 7 concludes.

2 Vessel Stowage Model

A Vessel Stowage Model (VSM) is a set of linear inequalities over continuous decision variables (i.e., a polyhedron) defining feasible stowage conditions of a container vessel. The VSMs in this paper are based on previous work on stowage planning optimization (e.g., [21, 12, 2, 16]).

Consider the container vessel shown in Figure 1. Each *cell* on the vessel can hold two 20' containers or one 40'. Some cells have power plugs, allowing refrigerated containers (*reefers*) to be stowed. Each container stack rests on sockets with maximum weight limits. Stacks are arranged longitudinal in *bays* that can be further subdivided into *locations*. The volume capacity of a vessel is measured in Twenty-foot Equivalent Units (TEUs) and can be more than 20K. Stowage conditions must satisfy a large number of interacting requirements. The vessel must be seaworthy with proper transversal stability and stress forces within limits. To that end, it can fill large water ballast tanks to achieve stability. The container stacks must by physically possible (e.g., 20' containers cannot be stowed on top of 40' containers) and there are separation rules for dangerous cargo. For an in-depth coverage of container vessel stowage, we refer the reader to a recent book on the topic [8].

The VSM that we investigate is based on industrial data from a large carrier that define volume, weight, and reefer capacities for each location of the vessel.

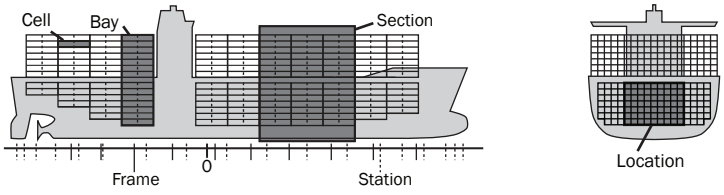

Fig. 1. Vessel structure and reference points.

Moreover, it gives positive and negative stress force limits (i.e., shear force and bending moment) for a set of *frame positions*. It also contains a Bonjean table that for different drafts gives the submerged area of a cross-section of the vessel for a number of *station positions* (see Figure 1).

The VSM considers 20' and 40' containers in three weight classes (6, 21 and 27 tons) and a container is either reefer and non-reefer. This gives total of 24 container types T. For each container type $\tau \in T$ and location l on the vessel, there is a decision variable $x_{l,\tau} \in \mathbb{R}$ defining the number of containers of this type in the location. Due to the large number of containers, we ignore the integrality of these variables as in [16]. The VSM includes volume, weight and reefer capacity constraints for each location. To simplify the representation of hydrostatic constraints, we divide the vessel into *sections* as shown in Figure 1. The VSM has constraints that restrict shear forces and bending moments to be within limits between sections. To this end it has constraints defining the resulting force on each section using a linear approximation of its buoyancy and weight. The weight of ballast water in each section s is given by a decision variable $t_s \in \mathbb{R}$. The hydrostatic modelling approach of the VSM is detailed in [9]. Due to space limitations, we do not include a formal definition of the VSM. It can be found in [1].

A Vessel Capacity Model (VCM) is derived from a VSM by adding auxiliary variables to the VSM equal to the total of each container type and projecting all other variables out using FME. In this way the container positioning information is abstracted away. While our experiments are carried out on the VSM from [1], we will for the sake of explaining our FME framework consider the toy version shown below exposing its block-angular structure.

$$S_{\mathtt{g}}: \begin{cases} x_\tau = \sum_{i \in S} x_{i,\tau} & \forall \tau \in T \quad (1) \\ \sum_{i \in S} \sum_{\tau \in T} W_\tau x_{i,\tau} \leq D & (2) \end{cases}$$

$$S_i \text{ for all } i \in S: \begin{cases} \sum_{\tau \in T^{\mathtt{20}}} x_{i,\tau} \leq C_i^{\mathtt{20}} \quad \sum_{\tau \in T^{\mathtt{20}}} W_\tau x_{i,\tau} \leq C_i^{\mathtt{W20}} & (3) \\ \sum_{\tau \in T \setminus T^{\mathtt{20}}} 2x_{i,\tau} \leq C_i^{\mathtt{40}} \quad \sum_{\tau \in T^{\mathtt{20}}} 0.5 W_\tau x_{i,\tau} + \sum_{\tau \in T \setminus \in T^{\mathtt{20}}} W_\tau x_{i,\tau} \leq C_i^{\mathtt{W40}} & (4) \\ \sum_{\tau \in T^{\mathtt{R}}} x_{i,\tau} \leq C_i^{\mathtt{RS}} \quad \sum_{\tau \in T^{\mathtt{20}}} x_{i,\tau} + 2 \sum_{\tau \in T \setminus T^{\mathtt{20}}} x_{i,\tau} \leq C_i^{\mathtt{TEU}} & (5) \end{cases}$$

In this toy VSM, the decision variable $x_{i,\tau} \in \mathbb{R}$ is the number containers in section i of type τ. There are two global constraints (1) and (2). The first defines the auxiliary variables x_τ that totals the container types (i.e., the only variables left in the VCM). The second is an assumed maximum weight D of the cargo, where W_τ is the weight of type τ. The following three constraints are defined for each section $i \in S$ and forms a block S_i in the block-angular structure. The first and second (3 and 4) are volume ($C_i^{\mathtt{20}}$ and $C_i^{\mathtt{40}}$) and weight capacity ($C_i^{\mathtt{W20}}$ and

$C_i^{\texttt{W20}}$) constraints of 20' and 40' containers, respectively. Notice that the weight limits of 40' containers includes half of the weight of 20' containers due to the arrangement of sockets. The last constraint (5) represents reefer ($C_i^{\texttt{RS}}$) and total TEU capacity ($C_i^{\texttt{TEU}}$). Notice that a 40' container counts two TEU.

3 Definitions and notation

A constraint system S is a set of equalities and inequalities over the same set of variables, $VAR(S) = \{x_1, \ldots, x_n\}$. Each constraint c is either an equality, written $a_1x_1 + \ldots + a_nx_n = b$, or an inequality, written $a'_1x_1 + \ldots + a'_nx_n \leq b'$. Alternatively, we use dot-product notation for the left-hand-side, i.e. $\mathbf{a} \cdot \mathbf{x}$. We let $var(c)$ denote the variables whose coefficient in c is nonzero and say that c *uses* x if $x \in var(c)$. The set of points in $\mathbb{R}^{|VAR(S)|}$ that satisfies all constraints in S is called the *feasible area* of S. A constraint $c \in S$ is *redundant* iff it does not influence the feasible area for S, otherwise it is called *non-redundant*.

For some variables $Y \subseteq VAR(S)$, we are not interested in their values in a feasible point - we just want to know that a satisfying value exists. This property is captured by the *projection of S w.r.t. Y*, which is the largest set consisting of values for $VAR(S) \setminus Y$ that can be extended with values for Y such that all constraints in S are satisfied. The projection of a constraint system is a uniquely determined subset of $\mathbb{R}^{|VAR(S)\setminus Y|}$, but also the feasible region of another system S' (see e.g. [22]). We are mostly interested in the latter, since it is the relationship between the values in the projection that is relevant to us, and we allow ourselves to write that "S' is the projection of S w.r.t. Y" if the feasible area of S' equals the projection of S, though such a system is not uniquely determined. We note, that since we are dealing with subsets of multi-dimensional Euclidian spaces, the dimension and the order of the variables are important. However, in order to simplify the presentation, we do not explicitly specify these for every considered projection/constraint system S'. A more stringent exposition keeping track of the variable sets and ordering can be found in [1].

4 FME-based projection framework

Our FME-based projection framework can be used for massive variable elimination in any linear inequality system but has been designed to take advantage of block-angular structures often found in real-world models including the VSM. The methods for projecting a constraint system are described in Section 4.1, while the decomposition used on block-angular structured problems is described in Section 4.2.

4.1 Projection procedure

The projection procedure starts with a preprocessing of the constraint system. Then we use the equalities in the reduced system to isolate variables from Y and

substitute in the rest of the system, i.e. *Gauss-elimination*. Subsequently, we successively eliminate one variable from Y using Fourier-Motzkin elimination and remove redundant inequalities. At the top-level, the pseudocode for our projection method is described in Algorithm 1. Each sub-procedure in this algorithm is detailed further below.

PREPROCESS(S, Y). We reduce S by removing easily identifiable redundant constraints and assign necessary bounds and values to variables using well-known LP preprocessing steps (e.g., [3]). The steps are implemented with special care of equalities and working with the assumption that the system is feasible (details can be found in [1]).

GAUSS-ELIM(S, Y). An equality e can be used to isolate a variable $x \in Y$ which can then be substituted in all other constraints in S (a Gauss-elimination). This eliminates x from the system and does not cause the same combinatorial explosion of inequalities as FME may do (e.g., [6, 19]). We Gauss-eliminate as many variables in Y as possible. To avoid density, when the system S contains several equalities, we first choose the variable x (used in any equality) that is used the fewest times in total in S. We then choose the equation e among those using x that uses the fewest variables, and do Gauss-elimination of x using e. This is then repeated until there are no more equalities using variables from Y.

FME-SINGLEVAR(S, Y). Fourier-Motzkin Elimination (FME) is a classical algorithm for producing the projection of a set of variables from an inequality system, i.e. a constraint system with no equalities. The method successively eliminates one variable $x \in Y$ until all required variables have been eliminated. To eliminate $x \in Y$, the constraints in S are first divided into three sets, $Pos_S(x)$, $Neg_S(x)$, and $Zero_S(x)$ depending on the sign of the coefficient of x. Each equality is treated as two inequalities, and bounds are treated as any other inequalities. A new system S' is then created, which is the projection of S w.r.t. $\{x\}$. It consists of $Zero_S(x)$, together with one inequality, $i_{p,n,x}$, for each pair $(p, n) \in Pos_S(x) \times Neg_S(x)$. $i_{p,n,x}$ is the addition of positive multiples of $p : \mathbf{a} \cdot \mathbf{x} \leq b$ and $n : \mathbf{a}' \cdot \mathbf{x} \leq b'$ such that the coefficient of x in the resulting inequality is 0, i.e.

$$i_{p,n,x} : -a'_x \cdot \mathbf{a} \cdot \mathbf{x} + a_x \cdot \mathbf{a}' \cdot \mathbf{x} \leq -a'_x \cdot b + a_x \cdot b'.$$

The order in which variables are eliminated naturally influences the size of the intermediary inequality systems. We have chosen to use the greedy heuristic that minimizes the number of new inequalities in the immediately next system

Algorithm 1 Projection based on Fourier-Motzkin elimination

```
function PROJECT(System S, Variables Y)
    (S, Y) ← PREPROCESS(S, Y)
    (S, Y) ← GAUSS-ELIM(S, Y)
    while Y ≠ ∅ do
        (S, Y, New) ← FME-SINGLEVAR(S, Y)
        S ← REMOVEREDUNDANCY(S, New)
    return S
```

[6], which is a commonly used heuristic. It is easily calculated from the current system as the variable $x \in Y$ that minimizes $|Pos_S(x)||Neg_S(x)| - |Pos_S(x)| - |Neg_S(x)|$. In the worst case scenario, the number of inequalities in the created system S' is $\frac{1}{4}|S|^2$, which implies that (both time and space) complexity is double-exponential. For a large, dense system, the growth will be substantial, which prohibits it from use for practical purposes *if* the added inequalities are non-redundant or the non-redundant inequalities are not removed (see e.g. [14]). It should, however, also be emphasized that not all inequalities in the succeeding system are necessarily non-redundant; in fact, the number of non-redundant inequalities will at most grow exponentially [15].

RemoveRedundancy(S, New). To detect redundancy, we examine each inequality $c : \mathbf{a} \cdot \mathbf{x} \leq b$ in turn and remove it from the system if $\max \mathbf{a} \cdot \mathbf{x}$ subject to $S \setminus \{c\}$ is less than or equal to b. The property can be checked using an LP solver. Only inequalities are examined, since we want to keep the equalities for use in Gauss-elimination. Not all inequalities have to be examined, though. When removing redundancy after projecting x from S, we do not need to check inequalities in $Zero_S(x)$; if they were non-redundant before the elimination, they will be non-redundant after. For large systems, checking all constraints for redundancy is time-consuming. We have therefore implemented a method for redundancy removal that uses several threads in parallel. Each thread checks one inequality at a time, while a manager takes care of the communication and keeps track of the found redundant inequalities. Several of the constants in the data used for our vessel models are results of various approximations and hence the boundary of the feasible area is not exact. Coarsening the boundary is therefore permissible, and we also remove inequalities that are *"almost redundant"*. An inequality $c : \mathbf{a} \cdot \mathbf{x} \leq b$ is almost redundant if $\max \mathbf{a} \cdot \mathbf{x}$ subject to $S \setminus \{c\}$ is less or equal to $b + \epsilon \cdot |b|$ for a small ϵ. Methods for coarsening the boundary of the feasible area are also used in [14, 18].

4.2 Decomposing a block-angular system

Looking at the simplified VSM from Section 2, it is clear that this has a natural (primal) block-angular structure ([20]): The set of capacity constraints for section $i \in S$ constitutes a local system, S_i whose constraints only use variables that are not used in $S_{i'}$ for a different section i'. The remaining constraints make up the global subsystem, $S_{\mathbf{g}}$, where the constraints also use variables from several of the local subsystems. This structure is illustrated in Figure 2(a) for a VSM with four sections.

To derive the VCM of this VSM, we want to eliminate all variables but $\{\ x^\tau \mid \tau \in T\ \}$, the variables counting the number of containers of each type. However, when a variable $x_{i,\tau}$ is eliminated, the inequalities constructed by FME uses variables from all subsystems, since $x_{i,\tau}$ is used in the global constraints. Continuing FME, this result in an increasing number of global and more dense constraints, which again makes FME perform worse. To avoid the immediate "mix" of local subsystems, we will exploit the system's block structure and define and use auxiliary variables to ensure that we can project the local subsystems

separately without producing global constraints, before we combine the projected subsystems and eliminate the auxiliary variables.

For the considered VSM, we first define a variable to hold the weight of cargo in each section i, w_i. We also define the variable $x'_{i,\tau} = x_{i,\tau}$, although this is only a renaming of the variables at first glance. We add the defining constraints to the relevant local system, and redefine the global constraints in terms of the new variables. That is, the i'th local subsystem is now $S_i^0 = S_i \cup \{w_i = \sum_{\tau \in T} W_\tau x_{i,\tau}\} \cup \bigcup_{\tau \in T} \{x'_{i,\tau} = x_{i,\tau}\}$, while the new system of global constraints is $S_g^0 = \cup_{\tau \in T} \{x^\tau = \sum_{i \in S} x'_{i,\tau}\} \cup \{w_1 + w_2 + w_3 + w_4 \leq D\}$.

Notice, that due to the new auxiliary variables, for a given $i \in S$, none of the variables $x_{i,\tau}$ is used in any constraints outside of S_i^0. Therefore, to project $\{\ x_{i,\tau} \mid \tau \in T\ \}$ from the whole system $(S_g^0 \cup_{i' \in S} S_{i'}^0)$ we can just eliminate $\{\ x_{i,\tau} \mid \tau \in T\ \}$ from S_i^0. In the end, when all subsystems S_i^0 have been projected, we can join these projections together with $S_{\mathfrak{g}}^0$ and eliminate the remaining variables, $\{\ x'_{i,\tau} \mid \tau \in T, i \in S\ \} \cup \{\ w_i \mid i \in S\ \}$, from the resulting system.

More formally, for a block-angular structured system with local sybsystems $S_1, \ldots, S_k$ (using the variables $X_1, \ldots, X_k$) and global subsystem $S_{\mathfrak{g}}$, to separate and remove local variables from the global constraints, we do as follows for all subsystems S_i (detailed pseudocode and correctness proofs can be found in [1]).

- For each global constraint c using variables in S_i, we define an auxiliary variable $z_{c,i}^0$ that equals the variables in S_i's contribution to c. We add the equality defining $z_{c,i}^0$ to S_i, and we substitute it in c. We name the thusly produced subsystem S_i^0.
- Then, we project S_i^0 w.r.t. all variables from $Y \cap X_i$, where $X_i = var(S_i)$, resulting in the system $S_i'^0$. We do keep the auxiliary z^0-variables. Because of these auxiliary variables this only produces inequalities with variables not present in other subsystems S_j.

After projecting each S_i^0 we can then combine the results with the rephrased, global constraints, $S_{\mathfrak{g}}^0$, to create the system $\mathcal{S} \stackrel{\text{def.}}{=} \mathcal{S} = S_{\mathfrak{g}}^0 \cup S_1'^0 \cup \ldots \cup S_k'^0$. We can then eliminate from $\mathcal{S}$ all the auxiliary z^0-variables, Z^0, plus any remaining variables in Y.

Comparing $\mathfrak{S} \stackrel{\text{def.}}{=} S_1^0 \cup \ldots \cup S_k^0 \cup S_{\mathfrak{g}}^0$ with the original system S, all we have done is defining auxiliary variables and substituted them in the system. Thus, eliminating Y from S is equivalent to eliminating $Y \cup Z^0$ from $\mathfrak{S}$. When eliminating $Y \cup Z^0$ from $\mathfrak{S}$, we can choose to first eliminate $X_1 \cap Y$, then $X_2 \cap Y$ up to $X_k \cap Y$, and finally $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$. Any variable in $X_1 \cap Y$ has a zero-coefficient in all constraints outside S_1^0, so $\mathfrak{S} \setminus S_1^0$ will not be changed by FME when $X_1 \cap Y$ is eliminated. It can therefore be put aside until that projection is done. Likewise, when eliminating $X_i \cap Y$, neither $S_{i+1}^0 \cup \ldots \cup S_k^0 \cup S_{\mathfrak{g}}^0$ nor the already projected systems contain any variables from $X_i \cap Y$ and can hence be put aside until the variables in $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$ are eliminated. Thus, the following holds.

Proposition 1. *The projection of S w.r.t. Y defines the same feasible area as the projection of $\mathcal{S}$ w.r.t. $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$.*

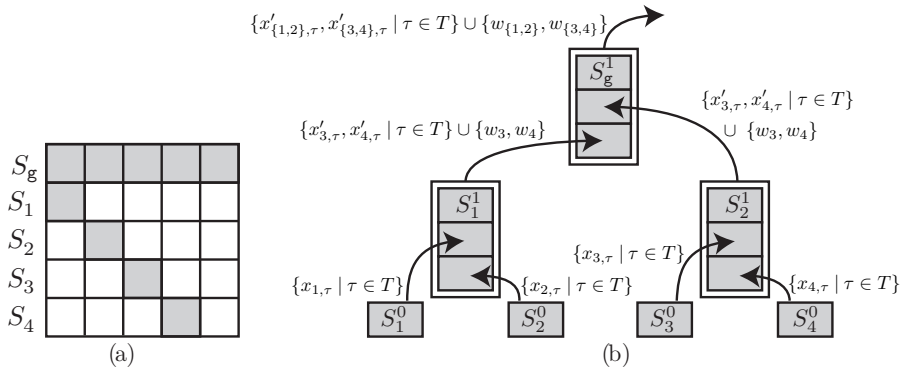

Fig. 2. (a) A block-angular structured system. (b) Projection using tree-structure.

$\mathcal{S}$ has by construction a block-angular structure and instead of eliminating $Z^0 \cup Y \setminus (X_1 \cup \ldots \cup X_k)$ immediately, if necessary, we can use the same approach as above to postpone "mixing" blocks. To do so, we collect all subsystems into k_1 small groups and do the following for each group i.

- We join the systems in the group into a new system, S^1_i. For each (rephrased) global inequality c using variables occurring in S^1_i, we then define a variable, $z^1_{c,i}$, that equals S^1_i's contribution to c. We add the defining equality to S^1_i and rephrase c using $z^1_{c,i}$.
- Then we project the resulting system, S^1_i, w.r.t. the previous, auxiliary z^0-variables, while we keep the newly created z^1-variables.

Subsequently we can then join the projected S^1-systems with S^1_g, and finally project the last auxiliary variables. Alternatively, we can repeat the steps above until the final projection can be done.

When we decompose a system as described above, we effectively create a tree structure of subsystem paired with a set of variables to be eliminated. An inequality system can then be projected by recursively projecting its children. See Figure 2 and the example further below. The intuition why this works is as before; projecting all Y and Z-variables from the union of all the (unprojected) subsystems in the nodes of the tree corresponds to projecting the Y variables from S, and because we can choose the elimination order of the variables, we only need to project the subsystems in the tree in the correct order (a rigorous proof can be found in [1]).

Proposition 2. *The projection of the system associated with the root of the tree constructed from S and Y w.r.t. the Y- and Z-variables as described corresponds to projecting S w.r.t. Y.*

Consider again the VSM that was decomposed into the subsystems S^0_g and S^0_i for $i \in S$. Instead of projecting these subsystems as described in the start of this section, we can insert an additional level in the decomposition, if for

examples the constraints in S_{g}^0 uses too many variables. Assume for instance that $S = \{1, 2, 3, 4\}$. Then we can group the four subsystems into two groups, $\{S_1^0, S_2^0\}$ and $\{S_3^0, S_4^0\}$, and for each global constraint, we define a variable stating each new group's contribution to the global constraint. For example, we define a variable $w_{\{1,2\}}$ that is the weight of the containers in section 1 and 2. The defining variables are added as a new subsystem, and the global constraints are rephrased. That is, we construct the following subsystems.

$$S_{\mathrm{g}}^1 : \{x^\tau = x'_{\{1,2\}} + x'_{\{3,4\}},\ w_{\{1,2\}} + w_{\{3,4\}} \leq D\},$$
$$S_1^1 : \{x'_{\{1,2\}} = x'_1 + x'_2,\ w_{\{1,2\}} = w_1 + w_2\},\ S_2^1 : \{x'_{\{3,4\}} = x'_3 + x'_4,\ w_{\{3,4\}} = w_3 + w_4\}.$$

These systems compose a tree structure, whose root is projected recursively as shown in Figure 2(b) to obtain the projection of the original VSM.

Using our previously described projection method, we obtain the projection of S w.r.t. Y by calling ProjectNode(root of T), where T is the tree structure constructed from S and Y, and ProjectNode is described in Algorithm 2. Using the described decomposition, it is of course also possible to project nested block structured problems, i.e. systems that on the top-level can be divided into a global part and a number of local parts that in themselves can be further divided into local parts and a global part, and so on. Other block structured problems such as staircase problems can also be decomposed into a tree structure and projected using the described approach.

Further, when the system S is decomposed into subsystems in a tree structure, the projection itself can be parallelized by maintaining a queue of not yet projected subsystems whose children have all been projected; this queue thus initially contains all leafs.

5 Results

We have constructed a number of different VSMs for a specific vessel, where the weight and hydrostatics are taken into account to various degrees. The first VSM has no limit on the total displacement or any hydrostatic constraints, the second VSM does not model any hydrostatic constraints, and the subsequent VSMs (refered to as complex VSMs) consider the stress force constraints at 2, 4, 6 and 8 measure points, respectively. Each VSM has been transformed into its corresponding VCM by eliminating all variables except the x_τ variables. Projections

Algorithm 2 Projecting a block-structured system via decomposition.

```
function ProjectNode(Node n)
    (S, Y) ← the system and variable set associated with n
    if n is a leaf then
        return Project(S, Y)                          ▷ Algorithm 1
    else
        for all children m of n do
            S ← S ∪ ProjectNode(m)
        return Project(S, Y)                          ▷ Algorithm 1
```

have been done in two different ways, *decomposed* and *flat*. For the decomposed projections, a tree structure has been used as described previously, while the flat projections do not use any decomposition at all. CPU time is measured in both number of iterations and *ticks* calculated by the CPLEX Interactive Optimizer version 12.5.0.0. Our FME-based projection framework has been implemented in Java. The experiments were carried out on a computer with an Intel® Xeon® CPU with 8 cores and 32GB RAM.

Table 1 shows the size reductions of the VCMs. It summarizes the size of the VSMs and the VCMs that are the result of the projection using decomposition. These sizes are given in terms of the number of inequalities (ineq), equalities (eq), variables (var), non-zero entries (nzs) and density (dens). The size of the VSMs are given both as they appear as input to our algorithm, but also after it has been preprocessed by CPLEX. For comparison, the table includes a "Simple VCM" corresponding to the maximum volume, weight, and reefer capacity models used in liner shipping today. Since we project all but 12 variables, this naturally gives a large reduction in the number of variables, more precisely 54-57 times fewer than even the presolved VSMs. However, the VCMs also have 5.8-11.8 times fewer inequalities than the presolved VSMs (for complex VSMs) while they two others have 20.8-27.7 times fewer constraints. The VCMs also have fewer non-zero entries (3-6 times fewer for complex VSMs, and otherwise 18-24). The results reveal no apparent relationship between the size of the VSM and the size of its VCM.

Regarding decomposition impact, Table 2 shows the time taken for the algorithm to do the projection, both decomposed and flat. For most VSMs, the flat projection timed out (TO) after 18 hours, in which case the variables left to be projected are given. Figure 3(a) shows the progression of the number of inequalities and variables, respectively, as a function of time when the algorithm runs on the decomposed 8 part model. These numbers are the sum of all the inequalities and variables, respectively, in all the projected or unprojected subsystems in the decomposition at a given time. Likewise, Figure 3(b) shows the progression for the flat projection of the same model; this figure includes the number of inequalities for the decomposed projection for comparison. Each graph shows the number of inequalities and variables after each step outlines in Section 4.1. The results in Table 2 shows that the decomposition has a substantial impact on the success of the projection of the complex VSMs. However, the two non-complex VSMs are solved faster using a flat projection. This is probably because they are

Table 1. The size of the VSMs and corresponding VCMs.

	VSM				VSM, presolved				VCM			
	ineq (eq)	var	nzs	dens	ineq	var	nzs	dens	ineq	var	nzs	dens
No weights	774 (12)	1142	6662	8.61	554	657	2784	5.03	20	12	155	7.75
No hydro.	806 (43)	1173	7854	9.74	555	657	3441	6.20	18	12	144	8.00
2 parts	810 (43)	1173	7860	9.70	556	661	3447	6.20	96	12	1113	11.59
4 parts	824 (49)	1179	7886	9.57	564	671	3471	6.15	64	12	731	11.42
6 parts	838 (55)	1185	7916	9.44	570	679	3496	6.13	80	12	888	11.10
8 parts	852 (61)	1191	7950	9.33	576	685	3522	6.11	52	12	582	11.19
Simple VCM	3	12	36	12.00	3	9	24	8.00				

Table 2. Projections time.

	Decomposed		Flat	
	time	vars left	time	vars left
No weights	24.5m	-	2.5m	-
No hydro.	14.5m	-	1.8m	-
2 parts	7h 18m	-	(TO) 32h	551
4 parts	8h 4m	-	(TO) 61h	557
6 parts	3h 7m	-	(TO) 18h	577
8 parts	3h 19m	-	(TO) 65h	566

sparse without the hydrostatic constraints.

When considering each subsystem in a decomposition as a system in itself, in general, the number of inequalities after each call to FME-SingleVar in Algorithm 1 grows to begin with, as does the number of inequalities before this call. This continues until there are a few variables left, where both these numbers decrease. For the decomposed algorithm, though the number of inequalities grow after each FME-step, most of them are redundant or almost redundant. The same does not hold for the flat projection of the complex VSMs (at least not for the FME-steps that are completed within the time limit). On the contrary, many of the produced inequalities are non-redundant, increasing the likelyhood that even more inequalities will be produced in the next elimination and that the redundancy removal will take longer time. We also note that the runtime, even for the decomposed projections, are not exactly small, and the main part of the execution time is spend doing redundancy removal. However, as mentioned in the introduction, these calculations only need to be done once per vessel class.

As a use-case example, the VSMs and their projected VCMs have been optimized for revenue. Each transported container yields a fixed revenue based on its type. Table 3 shows the number of iterations (iter), the deterministic time in ticks (time) and the optimal objective value (obj) in 10^6\$. It likewise shows how many times faster, the projections are w.r.t. iterations and deterministic time, as well as the difference in objective value in percentage. For comparison, the number of iterations, deterministic time and objective value is shown for the

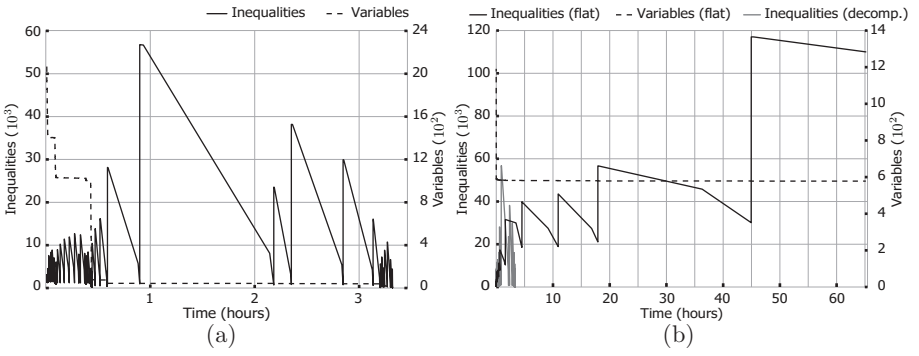

Fig. 3. The inequalities growth and variable decrease for the 8 part-model (a) with decomposition and (b) flat.

Table 3. Iterations, time and objective values for the VSMs and VCMs.

	VCM			VSM			Difference		
	iter	time	obj	iter	time	obj	iter	time	obj
No weights	11	0.05	8.63	363	2.64	8.08	×33.0	×52.8	6.8%
No hydro.	9	0.04	7.87	188	5.48	6.22	×20.9	×137	26.5%
2 parts	14	0.29	6.09	251	5.88	6.07	×17.9	×20.3	0.196%
4 parts	13	0.18	6.17	228	4.95	6.16	×17.5	×27.5	0.153%
6 parts	9	0.20	6.17	227	5.02	6.18	×25.2	×25.1	0.202%
8 parts	12	0.14	6.21	233	4.79	6.18	×9.4	×34.2	0.490%
Simple VCM	4	0.02	10.7						

simple VCM, too. As can be seen from the numbers in Table 3, in general, VCMs are much faster to solve than their corresponding VSMs. More specifically there are between approx. 17 and 33 times fewer iterations and 20-137 times fewer ticks, which corresponds to a difference between 94% and 97% of the number of iterations, and 96% and 99.5% CPLEX ticks, respectively. Meanwhile the difference in objective value is only modest; for the models including hydrostatic constraints, the difference is at most 0.5 %, while the other two models have a difference of 6.8 and 26.5 %, respectively. When comparing to the simple model, we see that this model of course is even faster (between 41-97 times (iterations) and 132-294 (ticks)), but the difference in objective is also between 72% and 76% for the last 5 models. Hence, our results confirm the experiments by Delgado [5] showing a substantial revenue overestimation of capacity models used in liner shipping today.

Beside the VSM, we have studied another block-angular structured system, namely one describing a multi-commodity flow problem. In short, this problem considers a graph on which a number of commodities can flow on the edges. Each edge has a capacity (upper bound) for each commodity as well as a common total capacity. Demands and supply are modelled as variables, and we want to examine the relationship between the supply and demand of the commodities without having to care about how the items flow in the internal nodes. This can be done by eliminating all other variables than the ones denoting the demands and supply of each commodity. We have generated the flow graph shown in Figure 4 with inspiration from the Chen.DSP collection [11]. It consists of seven "layers" with three nodes each, and there are two commodities. The capacity for each commodity and edge is 0 with a probability of 5% and otherwise drawn from a uniform distribution between 5 and 15, while the common capacity of the edge e is 0 with probability 25% and otherwise a number drawn from the uniform distribution. A multi-commodity flow problem is naturally block-structured with a block for each commodity, but it contains usually many global constraints. Therefore, instead of using these blocks to decompose the system, we divide the

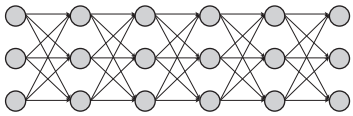

Fig. 4. A "layered" graph for a multi-commodity flow problem.

Table 4. Size of projection of a multi-commodity flow model.

	Size	Time
	ineq (eq) / var / nzs / dens	
Original	204 (42) / 120 / 444 / 2.17	-
Presolved	59 / 79 / 201 / 3.41	-
Projected, decomposed	17 (2) / 12 / 61 / 3.59	2h 9m
Projected, flat	17 (2) / 12 / 53 / 3.12	17h 41m

graph into smaller subgraphs. Similarly to Table 1, Table 4 shows the size of the original model and the projections resulting from a flat and decomposed projection, respectively. Figure 5 shows the progression over time of the number of inequalities and number of variables left to be projected, for both the flat and decomposed projection algorithm. Also here we see a reduction in the number of inequalities, variables and non-zero entries, of 3.5, 6.6, and 3.3/3.8 times, respectively (in both cases) compared to the presolved model. The density stays almost the same; for the decomposed model, the density increases with 5.3 %, while the density decreases with 8.5% for the flat projection. This is not as large a reduction as for the VSMs, however, the unprojected models are also smaller to begin with.

6 Related Work

Other FME-based frameworks for projection includes [19, 14, 18]. Similar to our framework, these use simplifications, redundancy removal and approximation-procedures. The latter uses the extreme-point method of [7], while the boundary-approximation of [14, 18] involves a successive increase of the allowable deviation from the feasible area and a permissible maximal ratio of removed non-redundant inequalities. Seen from the perspective of capacity models, both frameworks are used on quite small systems. Furthermore, the systems in [**?**] are also sparse. Our FME-based framework is to our knowledge the first that can take advantage of block-angular structure in the system. Other methods exist for computing the projection of a feasible area of an (in)equality system, that are not based on FME. The method in [7] finds extreme points in the projection space incrementally. It can therefore also be used to approximate the projection as is done in

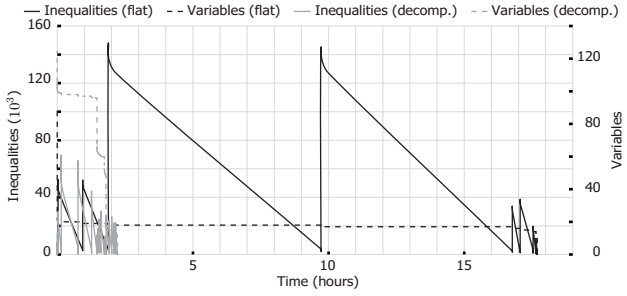

Fig. 5. Inequalities and variables during projection of a multi-commodity flow problem.

[19]. It is recommended for dense systems. Another example is the method introduced in [10], which computes all facets of the projection iteratively using a face-lattice. This method is recommended by the authors for polytopes with a low facet count and a high vertex count.

7 Conclusion

This paper has introduced a novel FME projection framework that automatically translates a linear stowage model (VSM) into a smaller sized capacity model (VCM) by projecting unneeded variables. To our knowledge, our framework is the first to exploit block-angular structure and apply massive parallelization of computations. Our results show that the projected VCMs are reduced by an order of magnitude both in number of inequalities and number of non-zero entries. The VCMs including hydrostatic constraints are solved 20-35 times faster than their corresponding VSMs. Similar results are achieved for at multi-commodity flow problem. Future work includes further parallelization and approaches to automatically estimates the best way of decomposing a given system as well as testing the framework on other block-angular problems.

Acknowledgements

We would like to thank Stefan Røpke, Thomas Stidsen, and David Pisinger for discussions on applications of the FME framework beyond container vessel capacity models. This research is supported by the Danish Maritime Fund, Grant No. 2016-064.

References

1. M. L. Ajspur and R. M. Jensen. Using fourier-motzkin-elimination to derive capacity models of container vessels. Technical Report TR-2017-197, IT University of Copenhagen, 2017.
2. D. Ambrosino, A. Sciomachen, and E. Tanfani. Stowing a containership: the master bay plan problem. *Transportation Research Part A: Policy and Practice*, 38(2):81 – 99, 2004.
3. E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71(2):221–245, 1995.
4. F. Benoy, A. King, and F. Mesnard. Computing convex hulls with a linear solver. *Theory and Practice of Logic Programming*, 5(1-2):259–271, 2005.
5. A. Delgado. *Models and Algorithms for Container Vessel Stowage Optimization*. PhD thesis, IT University of Copenhagen, 2013.
6. R. J. Duffin. *On Fourier's analysis of linear inequality systems*, pages 71–95. Springer, 1974.
7. T. Huynh, C. Lassez, and J.-L. Lassez. Practical issues on the projection of polyhedral sets. *Annals of mathematics and artificial intelligence*, 6(4):295–315, 1992.
8. R. Jensen, D. Pacino, M. Ajspur, and C. Vesterdal. *Container Vessel Stowage Planning*. Weilbach, 2018.

9. R. M. Jensen and M. L. Ajspur. The standard capacity model: Towards a polyhedron representation of container vessel capacity. In R. Cerulli, A. Raiconi, and S. Voss, editors, *Proceedings of the 9th International Conference on Computational Logistics (ICCL 2018)*, pages 175–190. Springer, Cham, 2018.
10. C. Jones, E. C. Kerrigan, and J. Maciejowski. Equality set projection: A new algorithm for the projection of polytopes in halfspace representation. Technical report, Cambridge University Engineering Dept, 2004.
11. K. Jones, I. Lustig, J. Farwolden, and W. Powell. Multicommodity network flows: The impact of formulation on decomposition. *Mathematical Programming*, 62:95–117, 1993.
12. J.-G. Kang and Y.-D. Kim. Stowage planning in maritime container transportation. *Journal of the Operational Research Society*, 53(4):415–426, 2002.
13. J.-L. Lassez. Querying constraints. In *Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 288–298. ACM, 1990.
14. A. M. Lukatskii and D. V. Shapot. A constructive algorithm for folding large-scale systems of linear inequalities. *Computational Mathematics and Mathematical Physics*, 48(7):1100–1112, 2008.
15. D. Monniaux. Quantifier elimination by lazy model enumeration. In T. Touili, B. Cook, and P. Jackson, editors, *Proceedings of the 22nd International Conference on Computer-aided verification*, pages 585–599. Springer Berlin Heidelberg, 2010.
16. D. Pacino, A. Delgado, R. Jensen, and T. Bebbington. Fast generation of near-optimal plans for eco-efficient stowage of large container vessels. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 286–301, 2011.
17. D. Pacino, A. Delgado-Ortegon, R. Jensen, and T. Bebbington. An accurate model for seaworthy container vessel stowage planning with ballast tanks. *Lecture Notes in Computer Science*, 2012.
18. D. V. Shapot and A. M. Lukatskii. Solution building for arbitrary system of linear inequalities in an explicit form. *American Journal of Computational Mathematics*, 2(01):1, 2012.
19. A. Simon and A. King. Exploiting sparsity in polyhedral analysis. In C. Hankin and I. Siveroni, editors, *Proceedings of the 12th International Symposium in Static Analysis (SAS)*, pages 336–351. Springer Berlin Heidelberg, 2005.
20. H. P. Williams. *Model Building in Mathematical Programming.* John Wiley & Sons, LTD, 1978.
21. I. D. Wilson and P. A. Roach. Container stowage planning: a methodology for generating computerised solutions. *Journal of the Operational Research Society*, 51(11):1248–1255, 2000.
22. G. M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate texts in mathematics.* Springer, New York, 1995.